The Processor: Datapath & Control

- We’re ready to look at an implementation of the MIPS
- Simplified to contain only:
  - memory-reference instructions: lw, sw
  - arithmetic-logical instructions: add, sub, and, or, slt
  - control flow instructions: beq, j

Regular Implementation:
- use the program counter (PC) to supply instruction address
- get the instruction from memory
- read registers
- use the instruction to decide exactly what to do

All instructions use the ALU after reading the registers

More Implementation Details

- Abstract / Simplified View:

  Two types of functional units:
  - elements that operate on data values (combinational)
  - elements that contain state (sequential)
State Elements

- Unclocked vs. Clocked
- Clocks used in synchronous logic
  - when should an element that contains state be updated?

![Clock waveform with falling and rising edges]

An unclocked state element

- The set-reset latch
  - output depends on present inputs and also on past inputs

![Set-reset latch diagram]
Latches and Flip-flops

- Output is equal to the stored value inside the element (don’t need to ask for permission to look at the value)
- Change of state (value) is based on the clock
- Latches: whenever the inputs change, and the clock is asserted
- Flip-flop: state changes only on a clock edge (edge-triggered methodology)

"logically true",
— could mean electrically low

A clocking methodology defines when signals can be read and written
— wouldn’t want to read a signal at the same time it was being written

D-latch

- Two inputs:
  – the data value to be stored (D)
  – the clock signal (C) indicating when to read & store D
- Two outputs:
  – the value of the internal state (Q) and it’s complement
D flip-flop

- Output changes only on the clock edge

Our Implementation

- An edge triggered methodology
- Typical execution:
  - read contents of some state elements,
  - send values through some combinational logic
  - write results to one or more state elements
Register file

Read register number 1
Read register number 2

Read data 1
Read data 2

Register file

Write

Register number
n-to-1 decoder

Register data

0
1
n
n – 1

Register 0
Register 1
Register n – 1
Register n
Register File

- Built using D flip-flops

Fetching instructions and incrementing PC
Datapath for R-type instructions

Datapath for load and store
Datapath for branching instructions

PC + 4 from instruction datapath

Add Sum Branch target

Shift left 2

ALU operation

To branch control logic

Instruction memory

Read address

Instruction

Read register 1

Read register 2

Read data 1

Read data 2

Write register

Write data

RegWrite

16

Sign extend

32

Building the Datapath

PCSrc

ALUSrc

ALU operation

Zero

ALU result

Shift left 2

Add ALU result

Add

PC

+ 4

Read data 1

Read data 2

Write data

Read register 1

Read register 2

Address

Data memory

MemRead

Sign extend

16

32

Sign extend

16
Complete Datapath